Final Report

Rohit

2023-11-26

Executive summary

Being tasked with the responsibility of analysing and building predictive models for Spotify as their Data scientist, various insights have been drawn in the process. The model was supposed to be built by variables like the genre of the song, the track release year, speechiness, the danceable and tempo characteristics of a song. The analysis also required asnwering a few questions which are asnwered as follows-:

1. The popularity did difffer between genres which is visualized by a box plot
2. Yes there is a difference in speeechiness between genres , the visualization depicts the clear difference , it is achieved through a box plot , another visualization was to calculate the mean distribution of speechiness and it is hence highlighted by the second bar graph.
3. Finally , yes the popularity did vary between genres which varied across the years the data is recorded.The visualization provides a clear picture of the variations.

Method

The data used in this analysis is a subset of the Spotify songs dataset from TidyTuesday. The data contains information about 32833 songs of different playlist/song genres on Spotify, including 23 other musical variables which give accurate insights about a specific song

1. Readin the data using readr package.
2. Selecting relevant columns
3. Omitting missing values
4. The year variable was generated by performing operations on release date variable 5.Data is sliced to 1000 rows per genre using appropriate functions
5. Factorization of variables is done.

Results

Model 1 (LDA)

The accuracy of model is 0.426 and 95%CI is 0.4008. The model seems fairly accurate but not the best.

Model 2 (KNN)

The K Nearest Neighbor (KNN) model had an accuracy of 0.476, i.e it predicted the correct genre of song 47% of time. The model performed little better than the LDA.

Conclusion

The data accuracy does not seem to be any high but it should be noted that predicting genre popularity based on the given features did not seem like a fair evaluation. Their is a significant overlap of the genre features. Nonetheless , our model still provides certain insights to the organization. The limitations of the model indicate that the organisation should incorporate additional relevant data for better prediction.

Appendix

Loading libraries

```
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 4.3.2
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidyr)
library(dplyr)
library(readr)
library(ggplot2)
library(lubridate)
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.3.2
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --
## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tune         1.1.2
## v infer        1.0.5     v workflows    1.1.3
## v modeldata    1.2.0     v workflowsets 1.0.1
## v parsnip      1.1.1     v yardstick    1.2.0
## v recipes      1.0.8
```

```
## Warning: package 'dials' was built under R version 4.3.2

## Warning: package 'infer' was built under R version 4.3.2

## Warning: package 'modeldata' was built under R version 4.3.2

## Warning: package 'parsnip' was built under R version 4.3.2

## Warning: package 'recipes' was built under R version 4.3.2

## Warning: package 'rsample' was built under R version 4.3.2

## Warning: package 'tune' was built under R version 4.3.2

## Warning: package 'workflows' was built under R version 4.3.2

## Warning: package 'workflowsets' was built under R version 4.3.2

## Warning: package 'yardstick' was built under R version 4.3.2

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

Data cleaning

```
spotify_songs <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/20
```

```
## Rows: 32833 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(spotify_songs)
```

```
## # A tibble: 6 x 23
##   track_id               track_name track_artist track_popularity track_album_id
##   <chr>                  <chr>      <chr>                   <dbl> <chr>
## 1 6f807x0ima9a1j3VPbc7VN I Don't C~ Ed Sheeran                 66 2oCs0DGTsRO98~
```

```
## 2 0r7CVbZTWZgbTCYdfa2P31 Memories ~ Maroon 5                67 63rPSO264uRjW~
## 3 1z1Hg7Vb0AhHDiEmnDE79l All the T~ Zara Larsson            70 1HoSmj2eLcsrR~
## 4 75FpbthrwQmzHlBJLuGdC7 Call You ~ The Chainsm~            60 1nqYsOef1yKKu~
## 5 1e8PAfcKUYoKkxPhrHqw4x Someone Y~ Lewis Capal~            69 7m7vv9wlQ4i0L~
## 6 7fvUMiyapMsRRxr07cU8Ef Beautiful~ Ed Sheeran              67 2yiy9cd2QktrN~
## # i 18 more variables: track_album_name <chr>, track_album_release_date <chr>,
## #   playlist_name <chr>, playlist_id <chr>, playlist_genre <chr>,
## #   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   duration_ms <dbl>
```

Overview of the data

```r
skim_without_charts(spotify_songs)
```

Table 1: Data summary

Name	spotify_songs
Number of rows	32833
Number of columns	23
Column type frequency:	
character	10
numeric	13
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
track_id	0	1	22	22	0	28356	0
track_name	5	1	1	144	0	23449	0
track_artist	5	1	2	69	0	10692	0
track_album_id	0	1	22	22	0	22545	0
track_album_name	5	1	1	151	0	19743	0
track_album_release_date	0	1	4	10	0	4530	0
playlist_name	0	1	6	120	0	449	0
playlist_id	0	1	22	22	0	471	0
playlist_genre	0	1	3	5	0	6	0
playlist_subgenre	0	1	4	25	0	24	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
track_popularity	0	1	42.48	24.98	0.00	24.00	45.00	62.00	100.00
danceability	0	1	0.65	0.15	0.00	0.56	0.67	0.76	0.98
energy	0	1	0.70	0.18	0.00	0.58	0.72	0.84	1.00

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
key	0	1	5.37	3.61	0.00	2.00	6.00	9.00	11.00
loudness	0	1	-6.72	2.99	-46.45	-8.17	-6.17	-4.64	1.27
mode	0	1	0.57	0.50	0.00	0.00	1.00	1.00	1.00
speechiness	0	1	0.11	0.10	0.00	0.04	0.06	0.13	0.92
acousticness	0	1	0.18	0.22	0.00	0.02	0.08	0.26	0.99
instrumentalness	0	1	0.08	0.22	0.00	0.00	0.00	0.00	0.99
liveness	0	1	0.19	0.15	0.00	0.09	0.13	0.25	1.00
valence	0	1	0.51	0.23	0.00	0.33	0.51	0.69	0.99
tempo	0	1	120.88	26.90	0.00	99.96	121.98	133.92	239.44
duration_ms	0	1	225799.81	59834.01	4000.00	187819.00	216000.00	253585.00	517810.00

Eliminating the null values

```
spotify_songs <- na.omit(spotify_songs)
```

```
song_select1 <-
  dplyr::select(spotify_songs, track_popularity, playlist_genre, track_album_release_date, danceability
  mutate(track_album_release_date = substr(track_album_release_date,1,4))

glimpse(song_select1)
```

```
## Rows: 32,828
## Columns: 6
## $ track_popularity         <dbl> 66, 67, 70, 60, 69, 67, 62, 69, 68, 67, 58, 6~
## $ playlist_genre           <chr> "pop", "pop", "pop", "pop", "pop", "pop", "po~
## $ track_album_release_date <chr> "2019", "2019", "2019", "2019", "2019", "2019~
## $ danceability             <dbl> 0.748, 0.726, 0.675, 0.718, 0.650, 0.675, 0.4~
## $ speechiness              <dbl> 0.0583, 0.0373, 0.0742, 0.1020, 0.0359, 0.127~
## $ tempo                    <dbl> 122.036, 99.972, 124.008, 121.956, 123.976, 1~
```

```
song_select1$year <- as.numeric(song_select1$track_album_release_date)
song_select <- dplyr::select(song_select1, track_popularity, playlist_genre, year, danceability, speechi
glimpse(song_select)
```

```
## Rows: 32,828
## Columns: 6
## $ track_popularity <dbl> 66, 67, 70, 60, 69, 67, 62, 69, 68, 67, 58, 67, 67, 6~
## $ playlist_genre   <chr> "pop", "pop", "pop", "pop", "pop", "pop", "pop", "pop~
## $ year             <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019,~
## $ danceability     <dbl> 0.748, 0.726, 0.675, 0.718, 0.650, 0.675, 0.449, 0.54~
## $ speechiness      <dbl> 0.0583, 0.0373, 0.0742, 0.1020, 0.0359, 0.1270, 0.062~
## $ tempo            <dbl> 122.036, 99.972, 124.008, 121.956, 123.976, 124.982, ~
```

Refactoring and slicing to 1000 rows each genre

```r
song_select <- song_select %>%
  group_by(playlist_genre) %>%
  sample_n(1000) %>%
  ungroup()
song_select
```

```
## # A tibble: 6,000 x 6
##    track_popularity playlist_genre  year danceability speechiness tempo
##               <dbl> <chr>          <dbl>        <dbl>       <dbl> <dbl>
##  1                0 edm             2013        0.601      0.0566  128.
##  2               45 edm             2014        0.413      0.0483  128.
##  3               14 edm             2018        0.805      0.085   124.
##  4               59 edm             2019        0.723      0.0354  128.
##  5               52 edm             2019        0.755      0.0381  122.
##  6               53 edm             2013        0.854      0.0494  120.
##  7                6 edm             2018        0.618      0.0882  148.
##  8               10 edm             2012        0.624      0.0621  128.
##  9               81 edm             2016        0.649      0.0349  100.
## 10               50 edm             2013        0.516      0.0538  124.
## # i 5,990 more rows
```

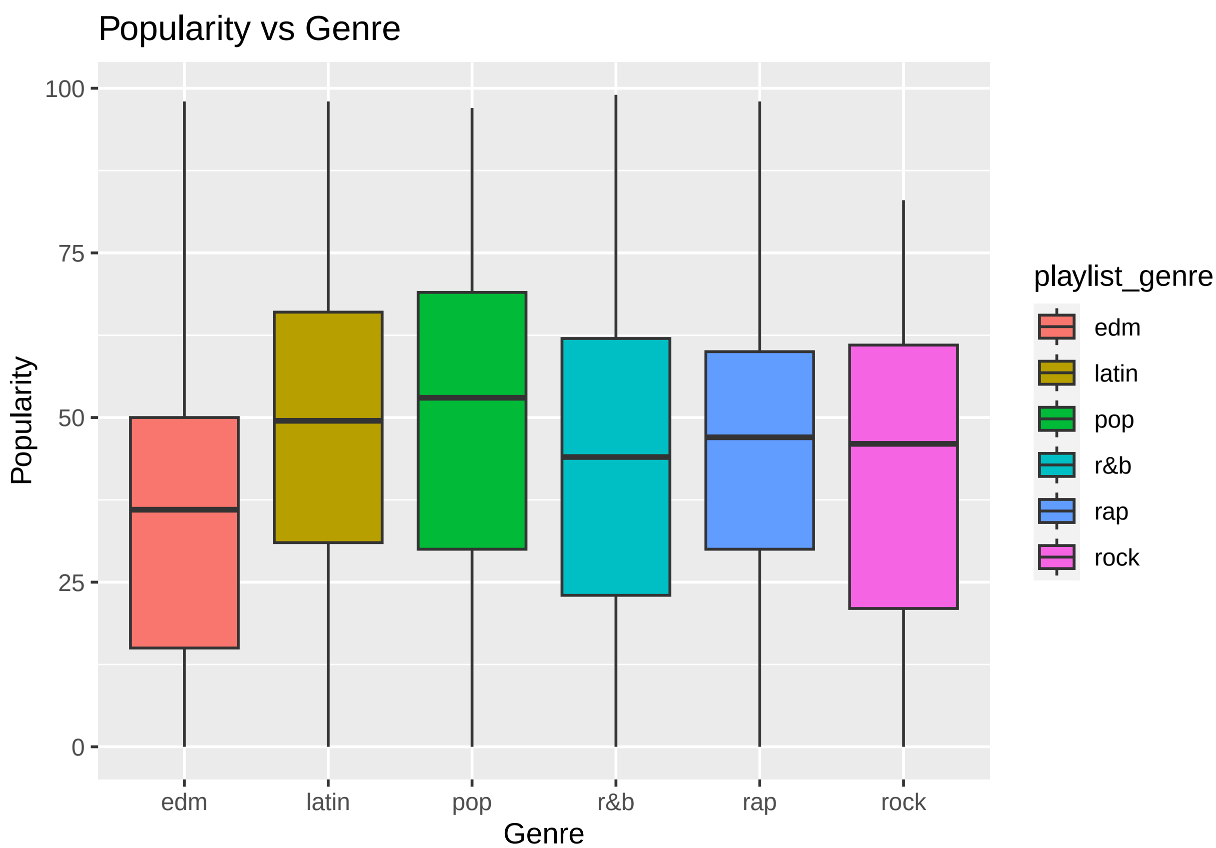
```r
song_select$playlist_genre = as.factor(song_select$playlist_genre)
glimpse(song_select)
```

```
## Rows: 6,000
## Columns: 6
## $ track_popularity <dbl> 0, 45, 14, 59, 52, 53, 6, 10, 81, 50, 2, 5, 6, 1, 0, ~
## $ playlist_genre   <fct> edm, edm, edm, edm, edm, edm, edm, edm, edm, edm, edm~
## $ year             <dbl> 2013, 2014, 2018, 2019, 2019, 2013, 2018, 2012, 2016,~
## $ danceability     <dbl> 0.601, 0.413, 0.805, 0.723, 0.755, 0.854, 0.618, 0.62~
## $ speechiness      <dbl> 0.0566, 0.0483, 0.0850, 0.0354, 0.0381, 0.0494, 0.088~
## $ tempo            <dbl> 127.992, 128.098, 123.995, 127.932, 121.992, 119.986,~
```

Exploratory analysis

Question 1

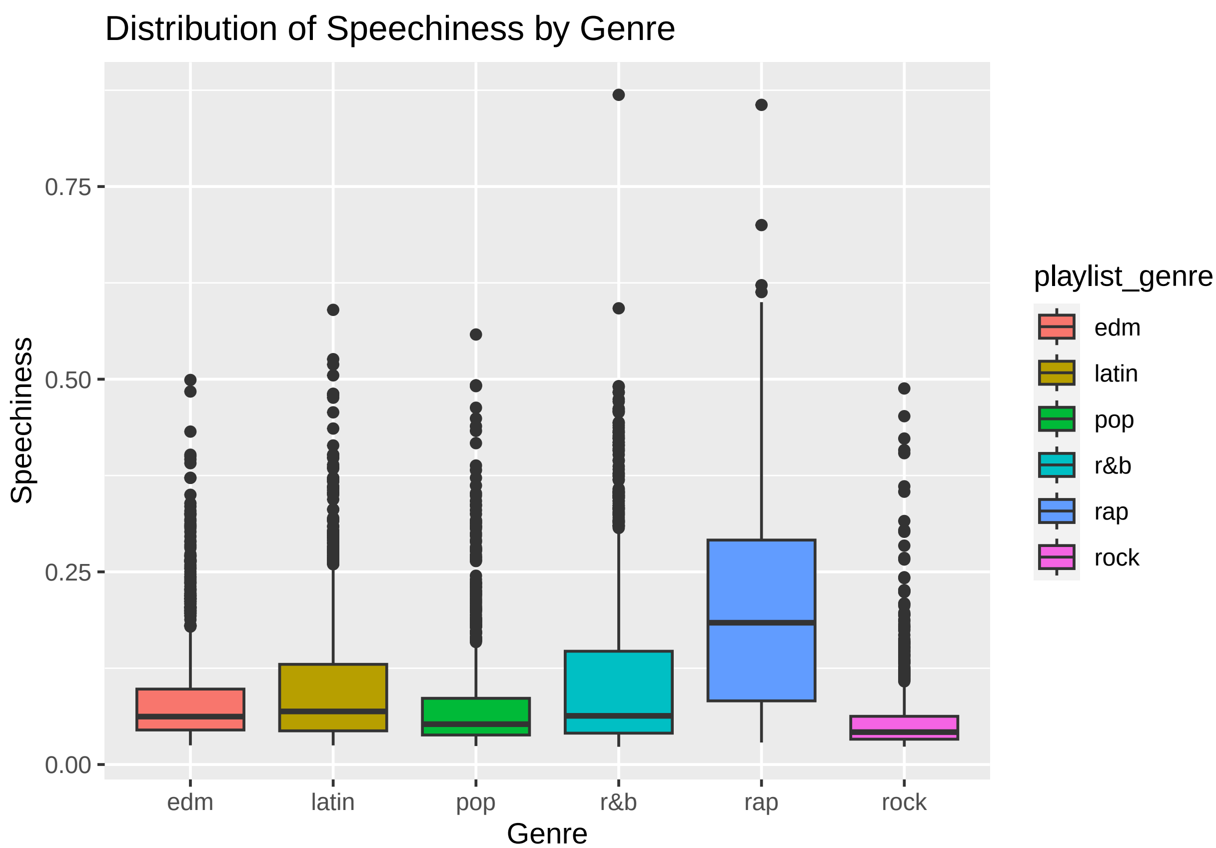
```r
ggplot(song_select, aes(x = playlist_genre, y = track_popularity, fill = playlist_genre)) +
  geom_boxplot() + labs(title = "Popularity vs Genre" , x = 'Genre', y="Popularity")
```



It is evident that 'Pop' has the most poluarity with 'Latin' slightly less popular than pop and rest of the genre having relatively high popularity than 'EDM' which is the least popular genre. It is thus concluded that popularity differs between genres.

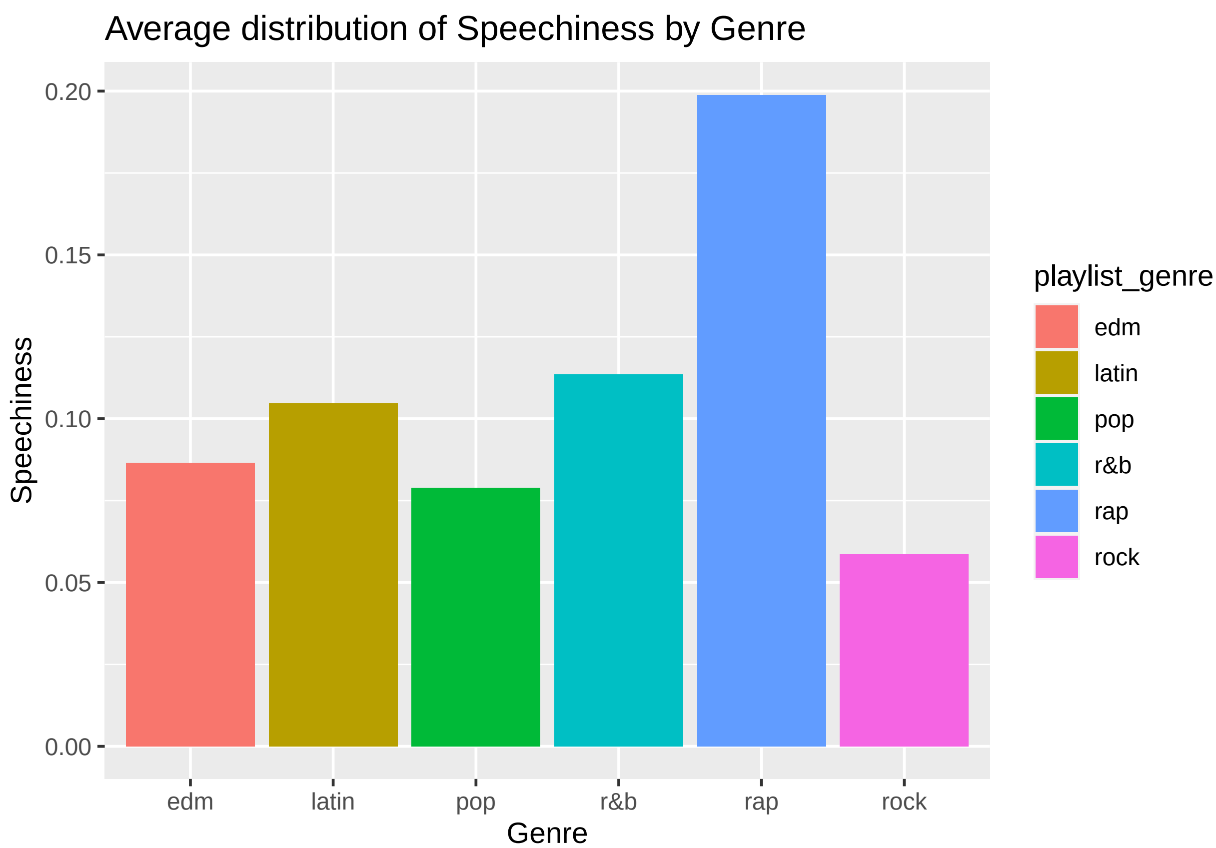
Question 2

```
ggplot(song_select, aes(x = playlist_genre, y = speechiness, fill = playlist_genre)) +
  geom_boxplot() +
  labs(x = "Genre", y = "Speechiness", title = "Distribution of Speechiness by Genre")
```



The above box plot concludes that 'Rap' and 'R&B' genre has more speechiness distribution than compared to the rest. 'Rock' genre being the least. We willl also try plotting against the mean distribution of speachiness against genre.

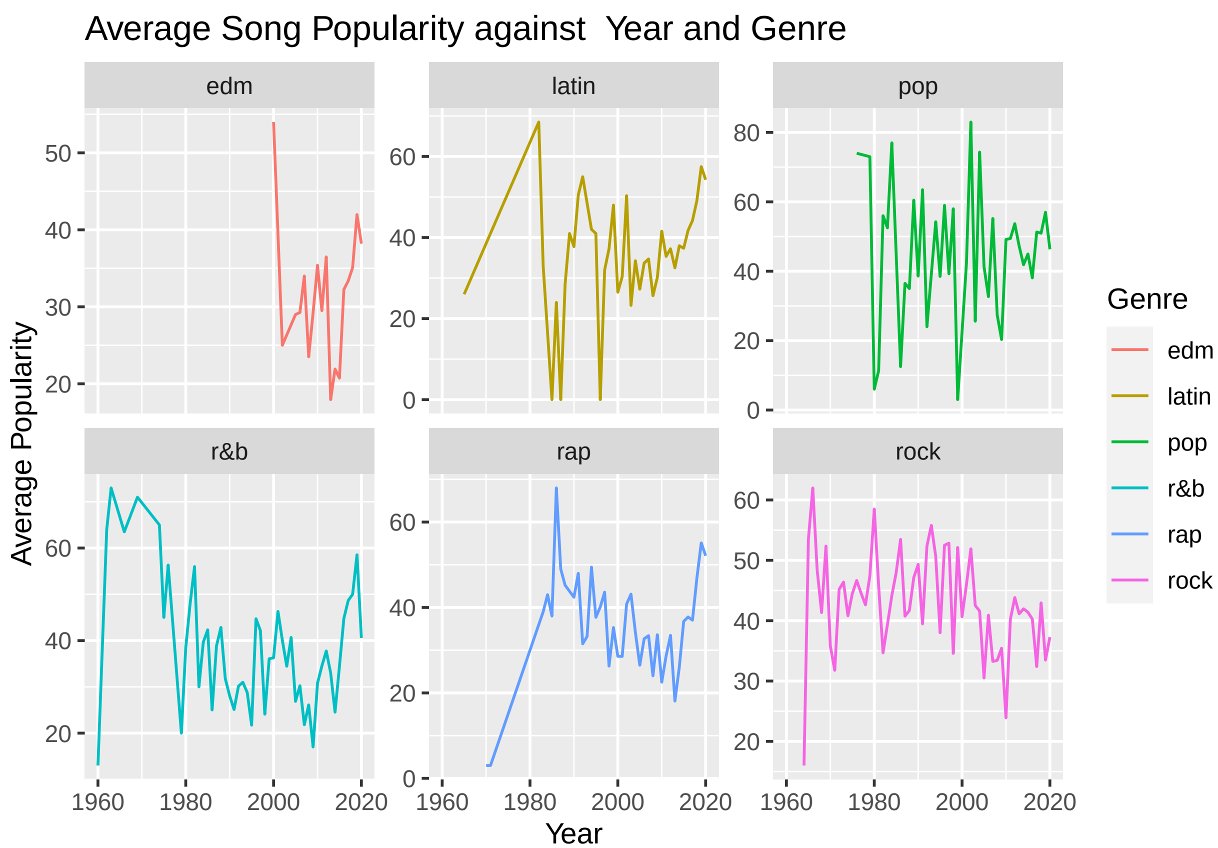
```
mean_speach<- song_select %>%
  group_by(playlist_genre) %>%
  summarise(avg = mean(speechiness) )

ggplot(mean_speach, aes(x = playlist_genre, y = avg, fill = playlist_genre)) +
  geom_bar(stat = 'identity') +
  labs(x = "Genre", y = "Speechiness", title = "Average distribution of Speechiness by Genre")
```



The second plot further strengthens the previous conclusion.

Question 3

```
song_select %>%
  group_by(playlist_genre, year = (year)) %>%
  summarize(avg_popularity = mean(track_popularity), .groups = 'drop') %>%
  ggplot(aes(x = year, y = avg_popularity, color = playlist_genre)) +
  geom_line() +
  labs(x = "Year", y = "Average Popularity", title = "Average Song Popularity against  Year and Genre")
  scale_color_hue(name = "Genre") +
  facet_wrap(~ playlist_genre, scales = "free_y")
```



The plot is the result of averaging popularity and grouping together with year and genre. SImple line plot is very complicated as trendlines overlap. The plot however potrays the average change in popularity of different genre over the time.

Model 1 -: Linear Discriminant analysis

Splitting and feature selection

```
set.seed(1893161)
song_split = initial_split(song_select)
song_train = training(song_split)
song_test = testing(song_split)
```

Generating the metrics for model 1

```
glimpse(song_train)
```

```
## Rows: 4,500
## Columns: 6
## $ track_popularity <dbl> 6, 34, 72, 45, 55, 0, 0, 37, 74, 47, 54, 74, 67, 54, ~
## $ playlist_genre   <fct> rock, rap, rock, latin, rap, rock, latin, rap, pop, r~
## $ year             <dbl> 1983, 2009, 1973, 2018, 2019, 1971, 2015, 2014, 2015,~
```

```
## $ danceability     <dbl> 0.536, 0.329, 0.429, 0.814, 0.933, 0.700, 0.478, 0.91~
## $ speechiness      <dbl> 0.0411, 0.0475, 0.0291, 0.0379, 0.0962, 0.0651, 0.398~
## $ tempo            <dbl> 166.766, 140.839, 136.302, 125.002, 125.009, 84.796, ~
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(recipes)
library(yardstick)
library(tune)
library(themis)
```

```
## Warning: package 'themis' was built under R version 4.3.2
```

```r
song_lda <- lda(playlist_genre ~ ., data = song_train)
song_pred <- predict(song_lda, newdata = song_test)
confusionMatrix(song_pred$class, song_test$playlist_genre)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction edm latin pop r&b rap rock
##      edm   136    49  58  34  39   43
##      latin  39    88  43  30  37    3
##      pop    55    66 100  53  25   29
##      r&b     4    15  12  32  12   10
##      rap    29    49  21  43 110    6
```

```
##      rock    3     9   23   41    8  146
##
## Overall Statistics
##
##                Accuracy : 0.408
##                  95% CI : (0.383, 0.4334)
##     No Information Rate : 0.184
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2877
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: edm Class: latin Class: pop Class: r&b Class: rap
## Sensitivity             0.51128      0.31884    0.38911    0.13734    0.47619
## Specificity             0.81929      0.87582    0.81657    0.95817    0.88337
## Pos Pred Value          0.37883      0.36667    0.30488    0.37647    0.42636
## Neg Pred Value          0.88606      0.85079    0.86604    0.85795    0.90258
## Prevalence              0.17733      0.18400    0.17133    0.15533    0.15400
## Detection Rate          0.09067      0.05867    0.06667    0.02133    0.07333
## Detection Prevalence    0.23933      0.16000    0.21867    0.05667    0.17200
## Balanced Accuracy       0.66528      0.59733    0.60284    0.54775    0.67978
##                      Class: rock
## Sensitivity              0.61603
## Specificity              0.93349
## Pos Pred Value           0.63478
## Neg Pred Value           0.92835
## Prevalence               0.15800
## Detection Rate           0.09733
## Detection Prevalence     0.15333
## Balanced Accuracy        0.77476
```

MOdel 2 KNN

```r
library(recipes)
song_recipe <- recipe( playlist_genre ~ ., data = song_train ) %>%
  step_downsample( playlist_genre ) %>%
  step_rm() %>%
  prep()
song_recipe
```

```
##
```

```
## -- Recipe ----------------------------------------------------------------------
```

```
##
```

```
## -- Inputs
```

```
## Number of variables by role

## outcome:   1
## predictor: 5

##

## -- Training information

## Training data contained 4500 data points and no incomplete rows.

##

## -- Operations

## * Down-sampling based on: playlist_genre | Trained

## * Variables removed: <none> | Trained
```

```
set.seed(1893161)
train_prep <- juice( song_recipe )
test_prep <- bake( song_recipe, song_test )
```

```
skim_without_charts(test_prep)
```

Table 4: Data summary

Name	test_prep
Number of rows	1500
Number of columns	6
Column type frequency:	
factor	1
numeric	5
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
playlist_genre	0	1	FALSE	6	lat: 276, edm: 266, pop: 257, roc: 237

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
track_popularity	0	1	42.32	25.13	0.00	24.00	46.00	62.00	98.00

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
year	0	1	2011.21	11.44	1965.00	2008.00	2016.00	2019.00	2020.00
danceability	0	1	0.66	0.14	0.08	0.56	0.67	0.76	0.97
speechiness	0	1	0.10	0.09	0.02	0.04	0.06	0.13	0.59
tempo	0	1	121.15	25.97	62.45	100.04	122.01	132.96	208.53

```r
knn_spec <- nearest_neighbor( mode = "classification", neighbors = tune() ) %>%
  set_engine( "kknn" )
```

```r
model_cv <- vfold_cv( train_prep, v = 5 ,strata = playlist_genre)
k_grid <- grid_regular( neighbors( range = c( 1, 100 ) ),
                        levels = 20)
```

```r
knn_tune <- tune_grid(object = knn_spec,
                      preprocessor = recipe(playlist_genre ~ .,
                                            data = train_prep),
                      resamples = model_cv,
                      grid = k_grid )
```

```
## Warning: package 'kknn' was built under R version 4.3.2
```

```r
best_acc <- select_best( knn_tune, "accuracy")
best_acc
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        58 Preprocessor1_Model12
```

```r
knn_spec_final <- finalize_model( knn_spec, best_acc )
knn_spec_final
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 58
##
## Computational engine: kknn
```

```r
spotify_knn <- knn_spec_final %>%
  fit( playlist_genre ~ . , data = train_prep )
spotify_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = playlist_genre ~ ., data = data, ks = min_rows(58L,     data, 5))
##
```

```
## Type of response variable: nominal
## Minimal misclassification: 0.5064457
## Best kernel: optimal
## Best k: 58
```

```r
knn_preds <- predict( spotify_knn,
                      test_prep,
                      type = "class" )
```

```r
head(tail(knn_tune %>%
            collect_metrics(),5))
```

```
## # A tibble: 5 x 7
##   neighbors .metric  .estimator  mean     n std_err .config
##       <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1        89 roc_auc  hand_till  0.798     5 0.00191 Preprocessor1_Model18
## 2        94 accuracy multiclass 0.491     5 0.00793 Preprocessor1_Model19
## 3        94 roc_auc  hand_till  0.798     5 0.00197 Preprocessor1_Model19
## 4       100 accuracy multiclass 0.490     5 0.00812 Preprocessor1_Model20
## 5       100 roc_auc  hand_till  0.798     5 0.00200 Preprocessor1_Model20
```